Arturo Zadot Aguilar J

Temario

1.1 Expresiones de Arboles
1.2 Acciones semanticos de un analizador
1.3 Comprobacion en tipos de expresiones
1.4 Pila semantica en analizador sintactico
1.5 Esquema de traduccion
1.6 Generacion de tabla y ambito y tabla de direcciones
1.7 Manejo de errores

2 Generacion de codigo intermedio

2.1 Notaciones
- 2.1.1 Prefija
- 2.1.2 Infija
- 2.2.3 Postfija

2.2 Representacion de codigo Intermedio
2.2.1 Notacion Polaca
2.2.2 Codigo P
2.2.3 Tripolos
2.2.4 Cuadripolos

2.3 Esquema de generacion
- 2.3.1 Variables y constantes
- 2.3.2 Expresiones
- 2.3.4 Instrucciones de control
- 2.3.5 Funciones
- 2.3.6 Estructuras

3 Optimizacion

3.1 Tipos de Optimizacion
3.1.1 Locales
3.1.2 Ciclo
3.1.3 Globales
3.1.4 Demirilla.

3.2 costo
3.2.1 Costos de ejecucion
3.2.2 Criterios de mejora al codigo
3.2.3 Herramienta de analisis para el flujo de datos

4 - Generacion de Codigo objeto

4.1 Registro
4.2 Lenguaje ensamblado
4.3 Lenguaje Maquina
4.4 Administracion de memoria

Notaciones

Infija { $y = 2x + 6$; $y = 2(x+3) - 5$

Postfija { 2, x, *, 6, +

2 x 3 + * , 5 , -

$f(t) = 3/(2+5)$.

El parentesis izquierdo siempre tiene la mayor prioridad.

Leemos la expresion de izquierda a derecha, si el elemento es un operando, entonces lo pasamos a la expresion postfija

Si el elemento es un operador se tendra que apilar, siguiente la regla numero 1. Si el elemento es un parentesis izquierdo lo trataremos como un operador. Aunque este no lo sea. Dicho tendra la mayor prioridad de todos.

Si el elemento es un parentesis derecho se procedera a apilar elementos en la pila hasta encontrar el elemento. Colocando cada uno de los elementos. Que estuvo desapilando en la expresion postfija

Regla #1:

Un operador se apilara. Siempre y cuando sea de mayor prioridad que el elemento en el elemento de la cima de la pila. Si no entonces quitaremos un elemento en la pila y volveremos a hacer el paso para apilar el operador.